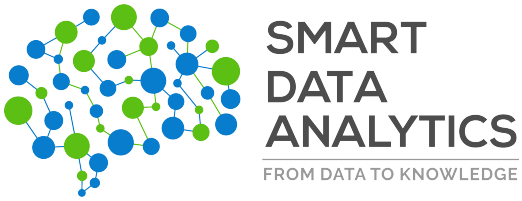

NATURAL LANGUAGE PROCESSING LAB

DEPARTMENT OF COMPUTER SCIENCE

SIMPLE QUESTION ANSWERING OVER DBPEDIA AND SUGGESTIVE QUESTION GENERATION

Mohnish Dubey

Chaitali Prabhu - 3069208
Debanjali Biswas - 3063994
Hemanth Kumar Reddy Mayaluru - 3063488
Kaushikee Khaund - 3068762

Contents

1 Introduction

Natural language processing (NLP) is a subfield of computer science, information engineering, and artificial intelligence concerned with the interactions between computers and human languages, in particular how to program computers to process and analyze large amounts of natural language data. Question Answering, which is a computer science discipline within the fields of information retrieval and natural language processing, is concerned with building systems that automatically answer questions posed by humans in a natural language. We develop a system to perform simple question-answering(questions with a single relation) on DBpedia, which returns the answer in a sentence and also generates a suggestive question.

2 Problem Statement

To develop a tool that takes a simple question (what, where, when) based on DBpedia triples as an input and gives two outputs.
1) Generates answer for the given question in a sentence.
2) Generates a suggestive question for the user.

3 Methodology

Given a natural language question,the workflow is divided into 4 parts:

- Entity Recognition and Entity Linking - detecting the entity in the question and getting the DBpedia url of the entity.
- Relation Detection, Answer Retrieval - detecting the direction and relation in the question and retrieving answer for the detected Entity and Relation by querying over DBpedia. Also, returning these answers as sentences.
- Semantic Coherence Calculation - From the 2 entities given in the answer, get the most relevant entity and from the predicates of this entity, choose a relation closest to the relation given in the answer.
- Generating Suggestive Question - Using the most relevant entity and closest relation, generate a suggestive question that the user can ask next.
  Each part is described in detail below. We consider the sample question:
  E.g. *who is the wife of Barack Obama ?*

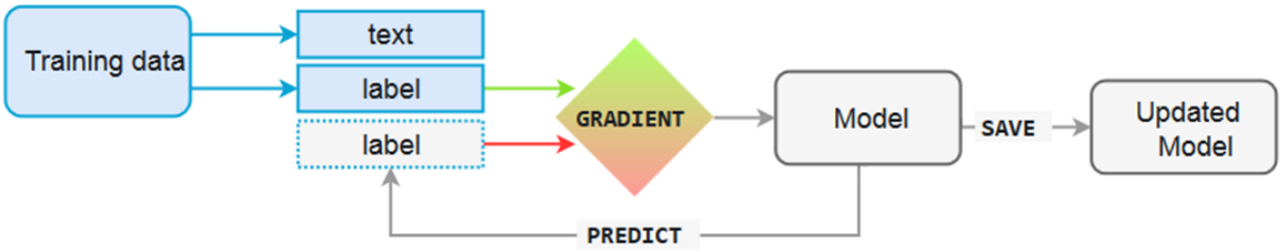

Figure 1: retrieved from: *https://spacy.io/usage/training*

3.1 Contributions

Member	Module Responsible
Entity Recognition and Entity Linking	Hemanth Kumar Reddy Mayaluru
Relation Detection, Answer Retrieval	Chaitali Prabhu
Semantic Coherence Calculation	Kaushikee Khaund
Generating Suggestive Question	Debanjali Biswas
Code Integration	Chaitali Prabhu Debanjali Biswas Hemanth Kumar Reddy Mayaluru Kaushikee Khaund
Front End	Chaitali Prabhu

3.2 Entity Recognition and Entity Linking

In this very first module, from a given question the task is to detect/identitfy the key entities and the relations.

3.2.1 Entity Detection

Entity detection popularly known as Named-entity recognition (NER) (also known as entity identification, or entity extraction) is a sub-task of information extraction that locate named entities from text into various categories such as the names of persons, monetary values, organizations, locations, quantities, percentages, etc.

For entity detection, a model has been created and trained over SimpleQA Dataset and wikidata. Using SPARQL queries, entity labels and types are extracted from DBpedia. Data preprocessing to make it available in the format needed to train the model. Spacy is used for model creation. spaCy's models(Figure 1) are statistical and every decision they make is a prediction. This prediction is based on the examples seen during the model training.The steps involve creation of a blank spacy model, training the model with the training dataset, saving the created model and testing it with the test

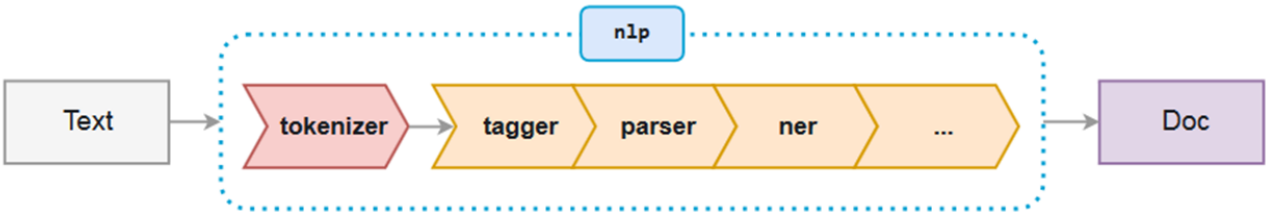

Figure 2: retrieved from: *https://spacy.io/usage/processing-pipelines*

dataset. Figure 2 shows the processing pipeline of the model which includes tokenizing the words, pos tagging, parsing and entity recognition.

SAMPLE TRAINING DATA:

[('which genre of album is harder faster ', 'entities': [(24, 31, 'Album')]), ('what movie is produced by Warner Bros ', 'entities': [(26, 37, 'Company')])]

The model is then shown the unlabelled text(in our case it's any question) and will make entity prediction. The model is given feedback on its prediction in the form of an error gradient of the loss function that calculates the difference between the training example and the expected output. The greater the difference, the more significant the gradient and the updates to our model.
We explained the inputs and outputs of the model in various stages using the question

Question: "who is the wife of Barack Obama?"

Model output: "Entity: Barack Obama"

3.2.2 Entity Linking

Data preprocessing is performed and Entity linking is done using EARL api which gives the relation from the question. Also using SPARQL the corresponding DBpedia URI is extracted based on the entity detected which will be one of the input to the next module of answer generation. The output of entity linking phase is as below:

Link: "http://dbpedia.org/page/Barack_Obama"

Relation: "wife"

3.3 Relation Detection, Answering Retrieval

This module includes detecting the direction of the question, detecting relation from the question, retrieving answers by querying over DBpedia and generating sentences for the answer.

3.3.1 Direction Detection

A given question can be forward or backward directed.
We use the training set (mentioned in the next section) and train a Logistic Regression. So, for a given question, we can predict if it is forward or backward directed.
Example for forward directed question: *who is the author of Farmer Boy ?*
Here, the entity will be detected as 'Farmer Boy' and the relation is detected as 'author'. So the triple would have the structure : <Farmer Boy><author><?answer>. In such cases, retrieving answers is very straight forward using SPARQL queries. Using the running example for this report: *who is the wife of Barack Obama ?* is a **forward** directed question. We only consider the forward directed questions in this project. Reason for not considering backward questions is explained below.
Example for backward direction question : *what is a book by Laura Ingalls Wilder ?*
Here, the entity will be detected as 'Laura Ingalls Wilder' which is correct. But, the relation will be detected as 'book by'. Since the question is backward, we would get a triple like: <?answer><book by><Laura Ingalls Wilder>which will not return any answer. Since the question is backward, we need the backward predicate for 'book by' as 'author' so the triple looks like: <?answer><author><Laura Ingalls Wilder>. Generating the backward predicates is not an option. We would need a dataset that has the pairs of predicates and the respective backward predicates. Therefore, retrieving answers for backward questions is out of the scope of this project.

3.3.2 Relation Detection

This part of the module deals with detecting the relation in the question which later is used to predict the predicate of the triple in DBpedia. We detect the relation by doing the following:

- Use nltk package to apply POS Tagging on the question. For the example: **[('who', 'WP'), ('is', 'VBZ'), ('the', 'DT'), ('wife', 'NN'), ('of', 'IN'), ('Barack', 'NNP'), ('Obama', 'NNP'), ('?', '.')]**
- If the question has verbs, they are considered for the relation building. If the question had verbs like 'directed', 'performed', etc., they would be used as the relation.
  For the example: Here the verb is **'is'**.
- If the question only has verbs whose infinite form would be 'be' or 'have', then we use the common nouns to form the relation. For the example: Since the question has the verb 'is', we would now consider the common noun 'wife' and together with the verb it would be considered as the relation.
  So the relation would now be **'wife is'**

We would have the entity, 'Barack Obama', and the url for the DBpedia page of the entity, 'http://dbpedia.org/resource/Barack_Obama', as output from the previous section. Using this, we make a list of all the forward predicates for 'Barack Obama' and choose the predicate which is semantically closest to the relation detected, using wordnet corpus. For the running example, the relation is detected as 'wife is' and from the list of predicates we would get the closest relation as **'spouse'**.

3.3.3 Retrieving answers from DBPedia

In this module, we retrieve answers from DBpedia by running a SPARQL query using the detected entity and relation. So the triple would be : **<Barack_Obama><spouse><?answer>**. Here, we get the answer **'Michelle Obama'**.

3.3.4 Generating answers as sentences

We maintain a file with a list of predicates and corresponding answer templates. For e.g.
1)weight (g), Y is the Z of X.
2)located in area, X is Z Y. and so on.. Finally, the X is replaced with the Subject, Y with the Object and Z with the Predicate. Also, for plural answers, the plural form of predicate is taken and 'is' is replaced by 'are'. For e.g.
writer, Y are the Z of X. (Z = writers)
performer, Y are the Z in X. (Z = performers)

3.4 Semantic Coherence Calculation

Semantic coherence is a notion introduced in Aronoff (1976) which entails that the meaning of a derivative is transparently a composition of the meaning of the base and that of the affix. We introduce the task of measuring semantic coherence in a question with respect to background knowledge, which relies on the identification of semantic relations between predicates in the knowledge graph (KG) and the predicate introduced in the question.
The aim of doing Semantic Coherence calculation is to use word embeddings to recognize relation in the question and determine their semantic closeness in a background knowledge graph. This will give us the relation which is *highly semantically closed* to the relation/predicate given in the question which will be used in the follow up question for the next part. The result from this part going to the next part would be an Entity and a Predicate.
This project in mainly focused on answering and generating questions with one triple (*s,p,o*).

3.4.1 Finding view counts of the Entities

In this part we take the two entities namely the subject and the object from the question and find the entity which has higher **View Counts**. View counts of the entities are taken from the Wikipedia data.
The Dataset used here is DBpedia. **SPARQL Query** is used to extract the labels from the URL of entities from DBpedia. Once we get the labels of the entities, we compare the view counts of each entity labels using the Wikipedia Rank Index files that we already have with us. The entity which has the higher view count is selected.
For the running example used here *who is the wife of Barack Obama ?*, we get two URL entities in the answer:
Entity1 = http://dbpedia.org/resource/Barack_Obama
Entity2 = http://dbpedia.org/resource/Michelle_Obama

After comparing view counts of the labels for these two entities, we get the view count of Entity1 to be higher. The output for this part is -
maximum index is for Entity1 http://dbpedia.org/resource/Barack_Obama

3.4.2 Generating all the Predicates for the Entity

For the entity with higher view count we find all the predicates from the KG of DBpedia. We sort all the predicates into two lists - Forward and Backward Predicates. **SPARQL Query** is used to extract the labels of predicates for each of the lists.

3.4.3 Finding Similarity of the predicates

In this part, from the set of all predicates we find the one which is the most semantically similar to the predicate given in the question.
Once we get the two lists of Predicate labels, we get the vector forms using **Spacy** for the predicate labels of each of the lists. Spacy has a number of different pre-trained models of word embeddings of different sizes available for use, with models in 7 different languages. Here, we have used the English model. We also find the vector form of the predicate given (*wife of*) in the question.
We use **Cosine Similarity Function** to find the similarity by measuring the cosine of angle between two vector forms of the predicates, one taken from the list of the predicates and the other given predicate.

$$similarity = \cos(\theta) = \frac{\mathbf{A}.\mathbf{B}}{||\mathbf{A}||.||\mathbf{B}||} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} \tag{1}$$

We compare the cosine similarity values for predicates of each of the lists and find the predicate which has the smallest distance to the given predicate. The predicate having the smallest distance is the one which is most similar to the given predicate in the question.
For the example taken in this paper *who is the wife of Barack Obama ?*, the most similar predicate to the given predicate in the question *wife of* is *Birth place*.

3.5 Generating Suggestive Question

In this module, we generate a simple suggestive question with the entity and predicate along with its direction received from the module before. The input to the model is a fact, consisting of subject, predicate and object, along with a question direction. We are using a **Seq2Seq** model for generating the questions as described in [3].
According to the example, the input to this module is a fact consisting of subject as http://dbpedia.org/resource/Barack_Obama, predicate as http://dbpedia.org/ontology/birthPlace and object as http://dbpedia.org/resource/Hawaii. It also receives the direction of the question as **forward**. The output from this model would be **Where was Barack Obama born?**

3.5.1 Seq2Seq Model for Question Generation

The seq2seq model consists of two parts. They are as follows:

- **Encoder:** The encoder encodes each atom of the fact into an embedding. A fact $F = \{s, p, o\}$, where s stands for subject, p for predicate and o for object are the individual atom, will be represented as a $1-of-K$ vector x_{atom}. The embedding is obtained by $e_{atom} = E_{in}x_{atom}$ where E_{in} is the embedding matrix of the input vocabulary and $E_{in} \in \mathrm{R}^{\mathrm{D_{enc}} \times \mathrm{K}}$ where K is the size of the input vocabulary. Now the encoder transform this embedding into $Enc(F)_{atom} \in \mathrm{R}^{\mathrm{H_{Dec}}}$ as $Enc(F)_{atom} = W_{Enc}e_{atom}$, where $W_{Enc} \in \mathrm{R}^{\mathrm{H_{Dec}} \times \mathrm{D_{Enc}}}$. Hence the *fact embedding* $Enc(F)_{atom} \in \mathrm{R}^{3\mathrm{H_{Dec}}}$ is the concatenation of $[Enc(F)_s, Enc(F)_p, Enc(F)_o]$. This fact embedding becomes the input to the next step.

- **Decoder:** The decoder is using a GRU recurrent neural network with attention mechanism. The decoder generates an associated question Q for a fact F according to the encoder representation. The hidden states of the GRU functions at every time step n as follows:

$$g_n^r = \sigma(W_r E_{out} w_{n-1} + C_r c(F, h_{n-1}) + U_r h_{n-1}) \tag{2}$$

$$g_n^u = \sigma(W E_{out} w_{n-1} + C c(F, h_{n-1}) + U_u h_{n-1}) \tag{3}$$

$$\overline{\mathrm{h}} = \tanh(W_r E_{out} w_{n-1} + C_r c(F, h_{n-1}) + U(g_n^r \circ h_{n-1}) \tag{4}$$

$$h_n = g_n^u \circ h_{n-1} + (1 - g_n^u) \circ \overline{\mathrm{h}} \tag{5}$$

where, σ is the sigmoid function, tanh is the hyperbolic tangent function, and $\circ$ is the element-wise multiplication. h_0, the initial state of the RNN, is the output from the encoder, i.e., fact embedding. $E_{out}w_n \in \mathrm{R}^{\mathrm{D_{Dec}}}$ is the decoder embedding of the word w_n, which is coded into a $1 - of - V$ vector, and E_{out} is the embedding matrix of the output vocabulary whose size is V. The variable U_r, U_u, U, C_r, C_u, $C \in \mathrm{R}^{\mathrm{H_{Dec}} \times \mathrm{H_{Dec}}}$, W_r, W_u, $W \in \mathrm{R}^{\mathrm{H_{Dec}} \times \mathrm{D_{Dec}}}$ are parameters of the GRU. The vectors g_n^r, g_n^u and $\overline{\mathrm{h}}$ are the *reset gate*, *update gate* and *candidate activation* respectively. $c(F, h_{n-1})$ is the context vector which is computed using an **Attention Mechanism** as given below:

$$c(F, h_{n-1}) = \alpha_{s,n-1} Enc(F)_s + \alpha_{p,n-1} Enc(F)_p + \alpha_{o,n-1} Enc(F)_o \tag{6}$$

where, $\alpha_{s,n-1}$, $\alpha_{p,n-1}$ and $\alpha_{o,n-1} \in \mathrm{R}$ are the weights which measure the contribution of the subject, predicate and object representations respectively.

3.5.2 Data Pre-processing

We have used the RevisedDBpediaQA (linked provided in section: 3), which is an extension of the SimpleDBpediaQA (linked also provided in section: 3). The dataset contains triples or facts, consisting of subject, predicate and object, direction, which is either forward or backward and questions. The pre-processing is done using the scripts provided in the link: https://github.com/castorini/SimpleDBpediaQA/tree/master/script. The pre-processing also involved removal of records containing abstract answers as objects.
The input and output vocabulary are created using *Gensim* [6].

3.6 Front End

Our module also has a simple front-end developed using *kivy* [8]. In the front-end, the first box is where we can type the question which we want to ask. To get the answer and the suggestive questions we need to press *Get Answer* button at the bottom of the page. The Answer and the Suggestive Question can be seen in the second and third box respectively. For out of vocabulary entities and predicates the system will print *"Entities or predicates out of vocabulary"* in the suggestive questions block.

4 Dataset and Evaluation

The knowledge base is DBpedia for all the modules. We have used https://raw.githubusercontent.com/castorini/SimpleDBpediaQA/master/V1/train.json and https://github.com/castorini/SimpleDBpediaQA/tree/master/revise_data_script as the training data. Every subsection below gives an evaluation based on the results of the respective module.

4.1 Entity Recognition and Entity Linking

Evaluation is done on both train and test dataset. We calculate the accuracy score, precision, recall and f-score for each entity that the model recognizes.Below are the values on the training and test dataset:

Dataset	Precision	Recall	f-score	Precision (Tokens)
Train	99.274	99.330	99.302	100.0
Test	84.467	83.041	83.748	100.0

$$precision = \frac{true\ positives}{true\ positives + false\ positives} \qquad recall = \frac{true\ positives}{true\ positives\ + false\ negatives}$$

$$F = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

Figure 1: Precision, Recall and f-score

4.2 Relation Detection, Answering Retrieval

Accuracy for:

- Logistic Regression used for direction prediction: 94.4970%
- Answer generation : 75-80%

Question	Answer	Result	Remark
who is the wife of Barack Obama ?	Michelle Obama is the spouse of Barack Obama.	Correct	-
what is the capital of Germany ?	what is the capital of Germany ?	Correct	-
what is the official language of India ?	Indian English, Standard Hindi are the languages of India.	Partially Correct	Both predicates(language and official language) at same distance, chooses the predicate that occurs first
which actors are starring in 3 Idiots ?	Generates no answer	Wrong	Treats as a backward question, can be dealt with more training data
what is the number of students in University Of Bonn ?	The number of students in University of Bonn are 32500.	Correct	-
what teams does Neymar play for ?	Neymar plays for teams Brazil national under-23 football team, FC Barcelona, Brazil	Correct	-
what is the currency of Philippines ?	Philippine peso is the currency of Philippines.	Correct	-
what is the height of Michael Jordan ?	The height (μ) of Michael Jordan is 1.9812.	Correct	-
what award has Tiger Woods won ?	Tiger Woods has won the awards PGA Tour, PGA Player of the Year, PGA Tour Player.	Correct	-
who was the founder of Samsung ?	Samsung is founded by Lee Byung-chul.	Correct	-
what is Ellen Swallow Richards's nationality ?	United States is the nationality of Ellen Swallow Richards.	Correct	-
what is the death place of He Dog ?	1930-1-1 is the death date of He Dog.	Wrong	If the exact relation is not found, the closest predicate is chosen
who directed the film Dhoom 2 ?	Sanjay Gadhvi is the film director of Dhoom 2.	Correct	-
who is the writer of The Grass Is Green ?	Mike Elizondo is the writer of The Grass Is Green.	Correct	-
what position does Graeme Murty play ?	The position of Graeme Murty is Defender (association football).	Correct	-

4.3 Semantic Coherence Calculation

Accuracy for a small dataset is found to be - 80%

4.4 Generating Suggestive Question

The training set contains 30,077 (after removal). We trained the model for 4 iterations with 20 epochs in each iteration. The final loss after training is 0.027343126. Figure 2 shows the learning curve for the first two iterations of training.

Figure 2: Learning Curve for Suggestive Question Generation

5 Conclusion and Future Work

We have developed a system that can generate answer and suggestive question for a given question. It works for all simple forward directed questions based on DBPedia. There are few shortcomings which are mentioned below, most of which can be solved with a bigger dataset.

5.1 Issue Faced

- Entity Recognition and Linking
  - Training the Model was taking too much time for each epoch.
  - After dividing the data into mini-batches, the speed of the model is increased.
- Relation Detection, Answering Retrieval
  - Generating answers for backward directed questions as mentioned in section 3.3.1
- Semantic Coherence Calculation

  - Problems faced for predicates having labels like *"Link to another Wikipedia page"* and other similar kinds of labels.

- Generating Suggestive Question
  - The model does not work for unseen predicates and out of vocabulary words. We can increase the size of the input vocabulary only if we have a larger dataset.
  - The dataset contains abstract answers. These were removed making the dataset even smaller.

5.2 Future Work

For each module, respectively,

1. Entity Detection and Linking
   - Train the model for still larger datasets
   - Entity Linking can be done for entities without capital letters
   - User friendly GUI can be developed
2. Relation Detection, Answering Retrieval
   - Train with more samples for direction prediction.
   - Use more efficient ways to compute similarity between predicates.
   - Add all predicates to list with sample sentence structure.
3. Semantic Coherence Calculation
   - Semantic Coherence can be done for more than two entities and also for complex questions
   - Generate more than one similar predicates (*e.g.* top five/top ten) for the entities
   - Consider all entities to generate their follow up questions instead of just taking one entity.
4. Generating Suggestive Question
   - Train the model for still larger datasets
   - Model can be updated for out-of-vocabulary words and unseen predicates
   - Generate multiple suggestive questions instead of one

References

[1] Dubey, M., Banerjee, D., Chaudhuri, D. & Lehmann, Jens. (2018). EARL: Joint Entity and Relation Linking for Question Answering over Knowledge Graphs.

[2] Salman Mohammed, Peng Shi, and Jimmy Lin. Strong Baselines for Simple Question Answering over Knowledge Graphs with and without Neural Networks. Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers), pages 291-296, May 2018, New Orleans, Louisiana.

[3] Serban, I., García-Durán, A., Gülçehre, Ç., Ahn, S., Chandar, A.P., Courville, A.C., & Bengio, Y. (2016). Generating Factoid Questions With Recurrent Neural Networks: The 30M Factoid Question-Answer Corpus. CoRR, abs/1603.06807.

[4] ElSahar, H., Gravier, C., & Laforest, F. (2018). Zero-Shot Question Generation from Knowledge Graphs for Unseen Predicates and Entity Types. NAACL-HLT.

[5] Britz, D., Goldie, A., Luong, M., & Le, Q.V. (2017). Massive Exploration of Neural Machine Translation Architectures. CoRR, abs/1703.03906.

[6] https://radimrehurek.com/gensim/index.html

[7] https://spacy.io/usage/processing-pipelines

[8] https://kivy.org/#home